Augmenting BiDAF with Components from R-NET for Question and Answering on SQUAD 2.0

Stanford CS224N Default Project
Track: SQuAD

Brandon Vu
Department of Computer Science
Stanford University
vubt@stanford.edu

Nyle Wong
Department of Computer Science
Stanford University
nylewong@stanford.edu

Richard Chen
Department of Computer Science
Stanford University
richchen@stanford.edu

Abstract

The Stanford Question Answering Dataset (SQuAD2.0) [1] is a reading comprehension dataset of question-passage pairs, where the answer to the question is some segment of the passage. In this paper, we use the Bidirectional Attention Flow (BiDAF) model [2] for question answering. We augment the BiDAF model with both gated attention and self-matching attention, as well as character embeddings as outlined in R-NET [3]. We experimented with different variations of character embeddings as well as hyperparameters for finetuning. We found the most optimal model was with gated-self-attention, 2 CNN character embeddings with kernal sizes 3 and 5, and with batch size of 72 and dropout of 0.1 throughout the model. This model achieved **F1: 65.869 and EM: 62.367** on the test set.

1 Key Information to include

- Mentor: Fenglu Hong
- External Collaborators (if you have any): None
- Sharing project: N/A

2 Introduction

Question answering is a central task in the application of deep learning to natural language processing models. For this task, models are fed some passage, known as the context, as well as some question, and must output some answer to this question, which can be found in the context. Question answering has been a prevalent area of natural language processing and deep learning research because question answering models necessitate the ability to process and formulate some sort of understanding of a given set of text. Outside of furthering potential knowledge in the natural language processing field, such models have applications to providing information to humans.

One dataset that has been used to evaluate question answering techniques is the Stanford Question Answering Dataset (SQuAD2.0) [1]. SQuAD2.0 consists of approximately 150,000 context-question-answer triplets, with the contexts drawn from Wikipedia and the question-answer pairs drawn from Amazon Mechanical Turk. For this dataset, the desired answer is certain span of words in the given context, so the question answering model is only expected to output a start word and end word of the

context. Notably, some questions in SQuAD 2.0 are "unanswerable": their answers cannot be found within the given context and the model is expected to indicate that this is so.

This paper presents our question answering model which manages to achieve a strong performance on SQuAD2.0 without the use of any pretrained transformers. The initial framework for our model is a Bidirectional Attention Flow (BiDAF) network [2], as outlined by Seo et al. To improve upon this baseline, we used both the self-matching attention and gated attention methods outlined in R-Net [3]. Additionally, we implemented character embeddings, which were obtained via convolutional neural networks. In testing, the best result of the model was an EM score of 62.367 and an F1 score of 65.869 on the test set , one of the strongest performances of any model that didn't use pretrained transformers or QANet architecture.

3 Related Work

The framework model and primary baseline for comparison we used was the Bidirectional Attention Flow network [2]. Initially introduced in 2016, this approach had three main contributions to machine comprehension attention models at the time. First, previous attention methods had encapsulated information from the context into a fixed size vector, while BiDAF does not fix this size and instead allows for vectors at all time steps and representations to "flow" into the next layer. Next, while most prior approaches calculated attention weights at each time step based on the vector at the previous time step, Seo et al. chose to design the attention to be temporally independent. Finally, they incorporated bidirectional attention from both context to query and query to context whereas previous models were only unidirectional. These modifications were hypothesized to improve performance via reducing information loss from early summarization as well as effective division of labor of corresponding tasks between the attention layer and the modeling layer. On the original SQuAD dataset, BiDAF managed to accrue single model scores of 68.0 and 77.3 and ensemble scores of 73.3 and 81.1 for the EM and F1 metrics, respectively.

Our approach incorporates the developments made in R-NET [3], a question answering neural networks model published in the following year. The R-NET model consists of two novel innovations. The first is the use of a gated attention recurrent network, which simply uses a gate to augment the input to the recurrent neural network (RNN) in that layer. This allows the model to pick out which parts of the context passage are most relevant to answering the given question. The second contribution is the self matching attention layer, which is implemented by having a gated attention bidirectional RNN run from the context to itself. The effect of this addition is to allow the passage representation to draw context clues and evidence from the entire passage rather than only from a narrow window. Together these modifications allowed R-NET to achieve single model scores of 72.3 and 80.7 and ensemble scores of 76.9 and 84.0 on the original SQuAD dataset, for the EM and F1 metrics respectively. Notably, these scores strictly outperform those achieved by BiDAF.

This paper uses character embeddings as well as R-NET's self-matching and gated attention to improve upon the starting BiDAF model. This is generally accepted to be the best possible performance of a model that does not utilize pretrained transformers or QANet architecture. Rather than make further improvement upon BiDAF + R-NET, we experiment and analyze the effects of the gated RNNs, kernel size and number of convolutional neural networks (CNNs) for character embeddings, and batch size and dropout on the effective performance of the model.

4 Approach

4.1 Baselines

We used the BiDAF model [2] with no modifications as the baseline model. We used the model provided by the sample code. We trained and evaluated this model on the dev set on which we got scores of **F1: 60.559 EM: 57.049**.

4.2 Self Attention

Our first approach to improve the BiDAF model was adding self-matching attention from the R-Net paper. This self attention allows the passage to be more aware of the context outside of its current

window. The attention is calculated as follows from the R-Net paper:

$$s_j^t = \mathtt{v}^{\mathtt{T}} \tanh(W_v^P v_j^P + W_v^P v_t^P)$$

$$a_i^t = \exp(s_i^t) / \sum_{j=1}^{n} \exp(s_j^t)$$

$$c_t = \sum_{i=1}^{n} a_i^t v_i^P$$

We take our context c_t and question-to-context embedding $h^{\tilde{}}$ and pass it through two distinct linear layers with no bias. We then apply the $\tanh$ activation and pass the result through another linear layer which has input size of the hidden size dimension and output size of 1. This produces s_t which represents the attention weights for a batch of sentences. We then take the Softmax over the sentence lengths to obtain a probability distribution of the attention weights, a_i^t. Finally, we take our attention distribution and apply it to the context, in this case $[v_t^P, c_t]$ where v_t^P or $h^{\tilde{}}$ is the query-to-context matrix. Our final attention output from self attention is $self\_att([u_t^P, c_t])$

4.3 Gated Attention

Our second approach to improving the BiDAF model was adding gated attention from the R-Net paper [3] to the BiDAF attention layer. The structure of the R-Net model is different than BiDAF. In the BiDAF model, the attention layer outputs $[h; u^{\tilde{}}; h \circ u^{\tilde{}}; h \circ h^{\tilde{}}]$ where h is the context and $u^{\tilde{}}$ and $h^{\tilde{}}$ are the context-to-query and query-to-context attention matrices respectively. In the R-Net paper, the attention output is $g \odot [u_t^P, c_t]$ where u_t^P is the context and c_t is the query-to-context attention matrix and g is the gate. However, instead of taking the gate over $[u_t^P, c_t]$, we apply it later to the entire attention output. We take $self\_att([u_t^P, c_t])$ and we concatenate the other attention matrices from the BiDAF model, namely $[a; h \circ u^{\tilde{}}; h \circ h^{\tilde{}}]$. In total, we are outputting an attention of $[self\_att([u_t^P, c_t]); a; h \circ u^{\tilde{}}; h \circ h^{\tilde{}}]$. The final step we take is applying a gate over this attention output, specifically the same gate, g as in the R-Net paper, a linear layer with no bias with a Sigmoid activation function applied over it.

4.4 Character Embeddings

In addition to tackling attention, we implemented character embeddings as according to the BiDAF model. These character level embeddings are obtained using Convolutional Neural Networks (CNN). The input size of the CNN is the size of the char embedding input vector, and the output size is the hidden size. We max pool over the maximum number of characters in a word. Then, we apply a $\tanh$ activation function after the CNN layer and the max pooling step to provide some non-linearity. This output is then concatenated with the word embeddings, and then the dimensions are projected back down to the size of the word embeddings, and activated again with $\tanh$.

Additionally, we added a more CNNs with a different kernel sizes in order to make the character embeddings more robust. The different kernal sizes represent different sized morphemes which we try to capture in the model.

4.5 Gated Recurrent Unit (GRU)

Like the R-Net paper, we also switched from an LSTM based model to using GRU. Making this choice early experimenting with our model, allowed us to reduce the training time significantly from 3 hours to 2 hours.

4.6 Finalized Model

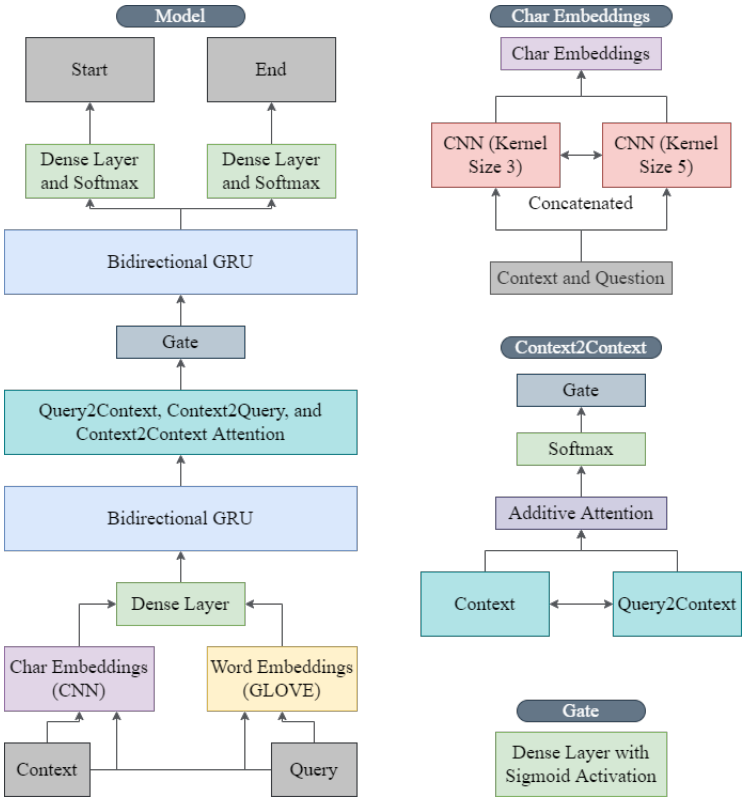


Figure 1: Model Diagram

5 Experiments

5.1 Data

The dataset that we are using is the Stanford Question Answering Dataset (SQuAD2.0) [1], which is a set of questions from Wikipedia articles where the answer is located within the text.

5.2 Evaluation method

The accuracy of the model on the SQuAD2.0 dev set was evaluated based on the exact match (EM) and F1 scores. The EM score is a binary metric that awards a score of 100% if the model's given answer exactly matches the desired answer, and 0% otherwise. The F1 score is a less strict metric that can be calculated as the harmonic mean of precision and recall. Precision is defined as the binary measure of whether the given answer is a subset of the desired answer, and recall is defined as the percentage of the desired answer's words that are present in the given answer. The model was evaluated on the dev and test set provided for the SQuAD2.0 dataset with these metrics.

5.3 Experimental details

All experiments were run on a Microsoft Azure VM. The hyperparameters used were the default values, with a hidden size of 100, a learning rate of 0.5, and the number of epochs as 30 with 50000 eval steps trained over the course of $\sim$2-3 hours each on the VM.

5.4 Results

Table 1: F1 and EM Results of Large Model Variations

Model Description	Dev Set F1	Dev Set EM	Test Set F1	Test Set EM
Baseline	60.559	57.049	60.811	57.143
+ GRU + Gated Self Attn	62.90	59.40	-	-
+ 1 CNN Char Emb (Kernal 3)	65.55	62.51	-	-
+ 2 CNN Char Emb (Kernals 2 & 3)	65.9	62.78	-	-
+ 2 CNN Char Emb (Kernals 3 & 5)	66.05	62.95	-	-
+ Increase batch size 64 -> 72	66.56	63.32	64.304	60.845
+ Decrease dropout probability to 0.1	**67.15**	**63.80**	**65.869**	**62.367**
+ Decrease dropout probability to 0	66.17	62.31	-	-
+ Increase batch size 72 -> 80	65.93	62.98	-	-
+ 2 CNN Char Emb (Kernals 2 & 5)	65.38	62.07	-	-
+ 3 CNN Char Emb (Kernals 2 & 3 & 5)	65.20	61.72	-	-

The best model was adding Gated Self Attention and 2 CNNs for character embeddings, specifically with kernal sizes 3 and 5, batch size of 72, and dropout of 0.1. The results are much better than we expected. We were surprised that adding 1 layer of character embeddings had a larger impact on the model than gated self attention. We hypothesize that this is because adding character embeddings helps the model understand morphenes in the English language which helps the model understand the question and context better than having a larger window to answer the question from self-attention.

5.5 Detailed Experiments

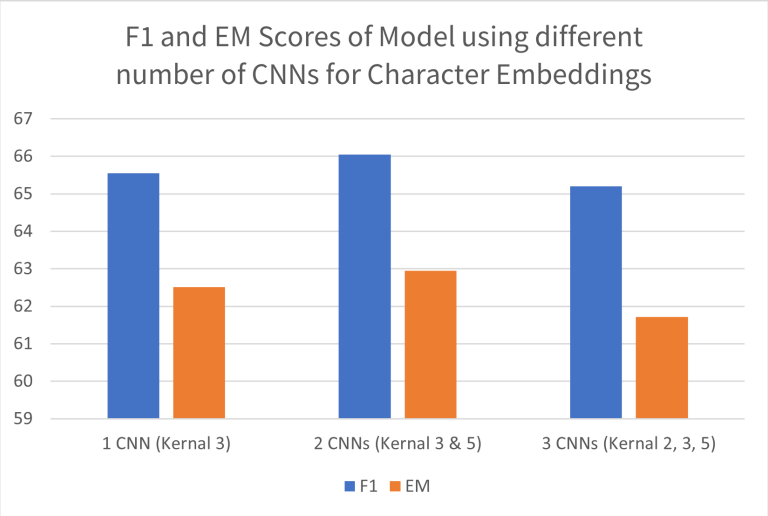


Figure 2: Bar graph of model performance when varying number of CNNs for Character Embeddings

The optimal kernal sizes were kernals 3 and 5. We seem to lose performance when adding 3 CNNs for character embeddings. This is likely because we are projecting down the entire concatenation of character and word embeddings to the same dimensions of the word embeddings. Therefore, the 3rd character embedding matrix probably caused the word embeddings to have less representation which led to decreased performance. We want to strike a balance between adding more kernal sizes to represent different sub-word lengths but also make sure that we aren't overpowering the word embeddings.

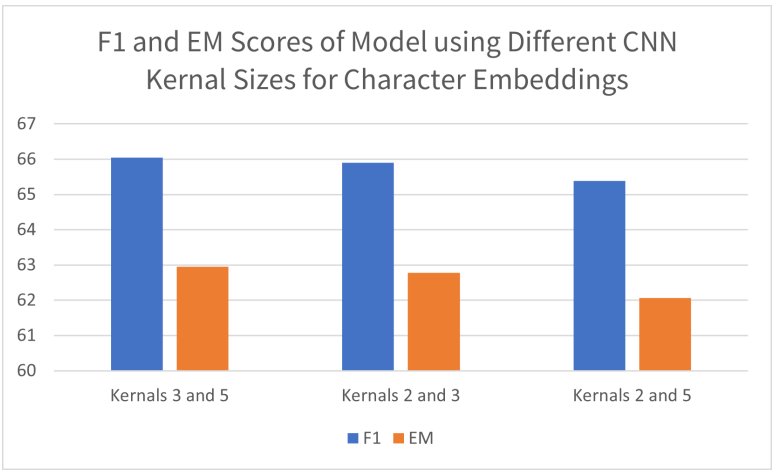
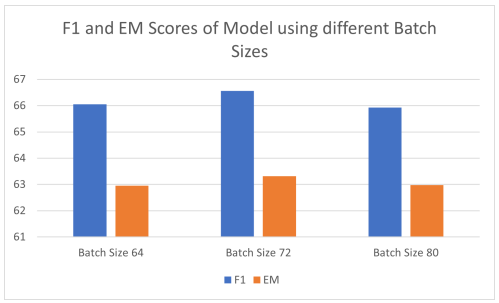
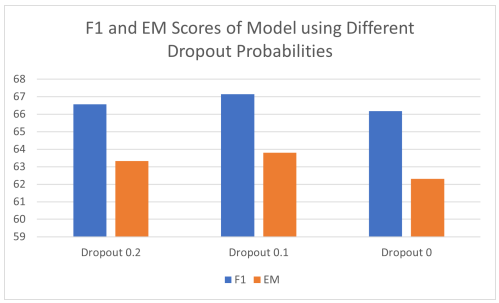


Figure 3: Bar graph of model performance when varying kernal sizes for 2 CNNs in Character Embeddings

We took the best model from the previous experiment and tried different kernal sizes used in the character embedding layer. From the results, the model with kernals 3 and 5 performed the best over the others. Something interesting we found was that kernals 2 and 5 did worse than the model with kernals 2 and 3. We would have thought that having kernal sizes too similar would lead to some redudancy over the sub-words that are being embedded, but it turns out that there might be more value to having 3 has a kernal size since it can embed a lot of common 3-letter prefixes and suffixes that is more valuable than a kernal size of 5.



(a) Varying Batch Size



(b) Varying Dropout

The batch size determines how much data is processed by the model before a parameter update. The more data we can pass in, the better performance on the training and validation set, but when the batch size is increased too much, we can lose performance on our validation set because we've overfitted on the train data. This trend is seen here where we get some improvement for increasing the batch size, but passed a batch size of 72, it seems we lose performance on the validation set. We also experimented with the dropout probability of the model.

Dropout is where we set certain parts of the model to 0 in order to build more robust connections between layers of the model. We found a similar trend to batch size with dropout where there is an optimum probability of dropout of 0.1 where we can maximize our validation and test set performance.

6 Analysis

Upon examination of specific outputs of the model to questions, we found the following characteristics.

First, we can categorize the questions into two large categories, one where the question is phrased in a more standard and grammatically simple manner, and another where the format of the question is more complex. For the sake of the analysis, question-word-first questions are defined as questions in the form where "what/when/why/where/how" comes at the start of the sentence, such as "what year did Barak Obama become President?" Subject-first questions are when the question-word is not at the beginning of the sentence, such as "Barak Obama became president in what year?"

Between these two types of questions, we observed that our model struggled significantly more with subject-first questions. More specifically, they were more likely to predict "N/A" despite there being an available answer. On the other hand, our model was able to more accurately predict the correct answer for question-word-first questions. Additionally, when there is no available answer in the context (when the correct answer is "N/A"), the model is more likely to erroneously make a false prediction for question-word-first questions compared to subject-first questions. We hypothesize that because the question-word-first questions are easier to predict in general, there is more potential for overconfidence in the predictions by the model.

Next, when comparing the outputs of the baseline model with our finalized model, we found that our model was stronger at correctly predicting "N/A" when there is no available answer. Our model was also able to match shorter and more concise phrases. For example, one question answer pair was as follows:

Question: Why did the oil ministers agree to a cut in oil production?
Answer: until their economic and political objectives were met

The answers predicted by the baseline and our model were as follows:

Baseline prediction: five percent monthly increments until their economic and political objectives were met
Model prediction: until their economic and political objectives were met

Our model was able to more accurately recognize the section of the context that matched what the question was looking for. We hypothesize that this is due to the addition of self-attention that helps the model interpret the context more accurately.

As a whole, we observed that both the baseline and our model generally struggled more with questions involving "why." In many instances, the predicted output would be "N/A" despite there being an available answer. This is likely due to the fact that "why"-questions are typically less straightforward and leave more up to the interpretation by the model. One such example is as follows:

Question: According to Lenin why must capitalistic countries have an imperialistic policy?
Answer: to constantly expand investment
Baseline prediction: N/A
Model prediction: N/A

For this question specifically, the fact that the question-word is not at the front could have made it even more difficult for the model to accurately predict.

Finally, we noticed that the models also generally performed well with questions that expected numbers as the answer, which could be due to the fact that numbers are easy to understand and interpret within the paragraph of context.

Overall, the our model showed concrete improvement over the baseline, and we have documented several areas that could be targeted for improvement.

7 Conclusion

Ultimately, we successfully implemented a model with character embeddings, as well as the self-matching attention and gated attention from R-NET, to achieve F1 and EM scores of 65.869 and 62.367 respectively on the test set. This outperformed the baseline BiDAF model scores of 60.811 and 57.143, for an increase of about 5 percent.

In addition to improving the model's scores, we were able to show that modifying certain characteristics of the system induced changes in the performance. Properties such as the number of CNNs used for the character embeddings, the CNN kernel sizes, batch size, and dropout probability all affected the accuracy of the model. In the end, of the ones we tested we found that using a batch size of 72 and 2 CNNs with kernel sizes of 3 and 5 maximized our performance.

The performance for this model is limited by nature of not utilizing QANet architecture or pretrained transformers. Even though we believe the performance of our model is close to the ceiling for its type, there are some possible further optimizations to be done. It is very possible that the kernel sizes, batch sizes, and dropout rates can be finetuned further for some slight improvements in performance. Another consideration for future work is the possible implementation of pointer networks, which modifies the neural attention mechanism to be used as a pointer to an input element instead of weights for the encoder outputs. Pointer networks can allow the network to learn structures where output sequence elements correspond to positions in the input sequence.

References

[1] Konstantin Lopyrev Pranav Rajpurkar, Jian Zhang and Percy Liang. Squad: 100, 000+ questions for machine comprehension of text. 2016.

[2] Ali Farhadi Minjoon Seo, Aniruddha Kembhavi and Hannaneh Hajishirzi. Bidirectional attention flow for machine comprehension. 2016.

[3] Microsoft Research Natural Language Computing Group. R-net: Machine reading comprehension with self-matching networks. 2017.